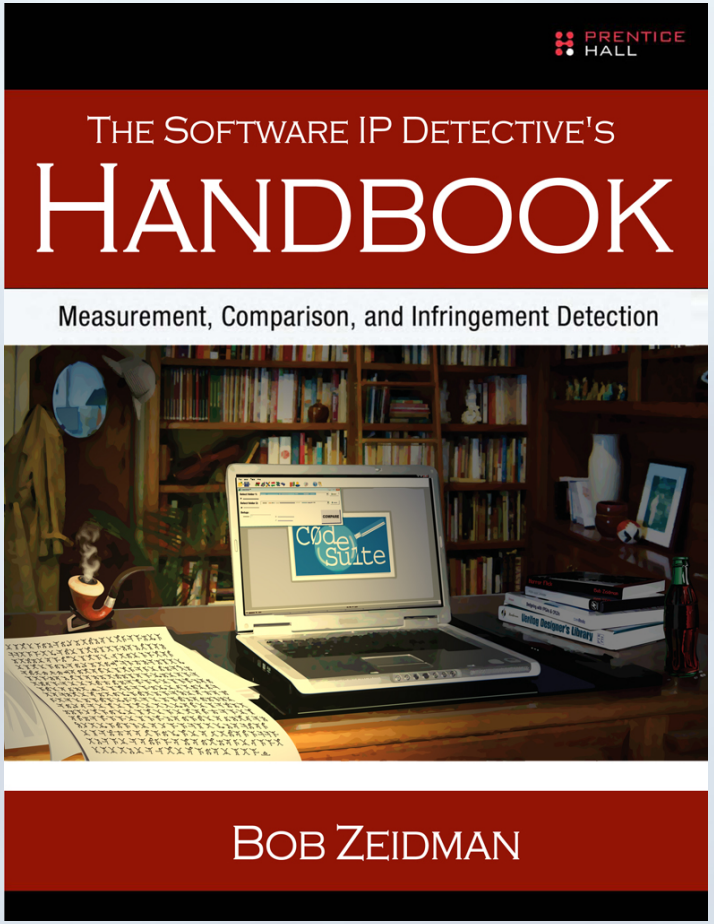
Detecting Software IP Theft

About Bob Zeidman



- President of Zeidman Consulting
- President of Software Analysis & Forensic Engineering Corporation
- Developer of CodeSuite®
- Author of *The Software Detective's Handbook*
- Famous cases:
  - *ConnectU v. Facebook*
  - *Symantec v. IRS*
  - *Oracle v. Google*
  - *Diyora & Bhanderi v Sarine Technologies*

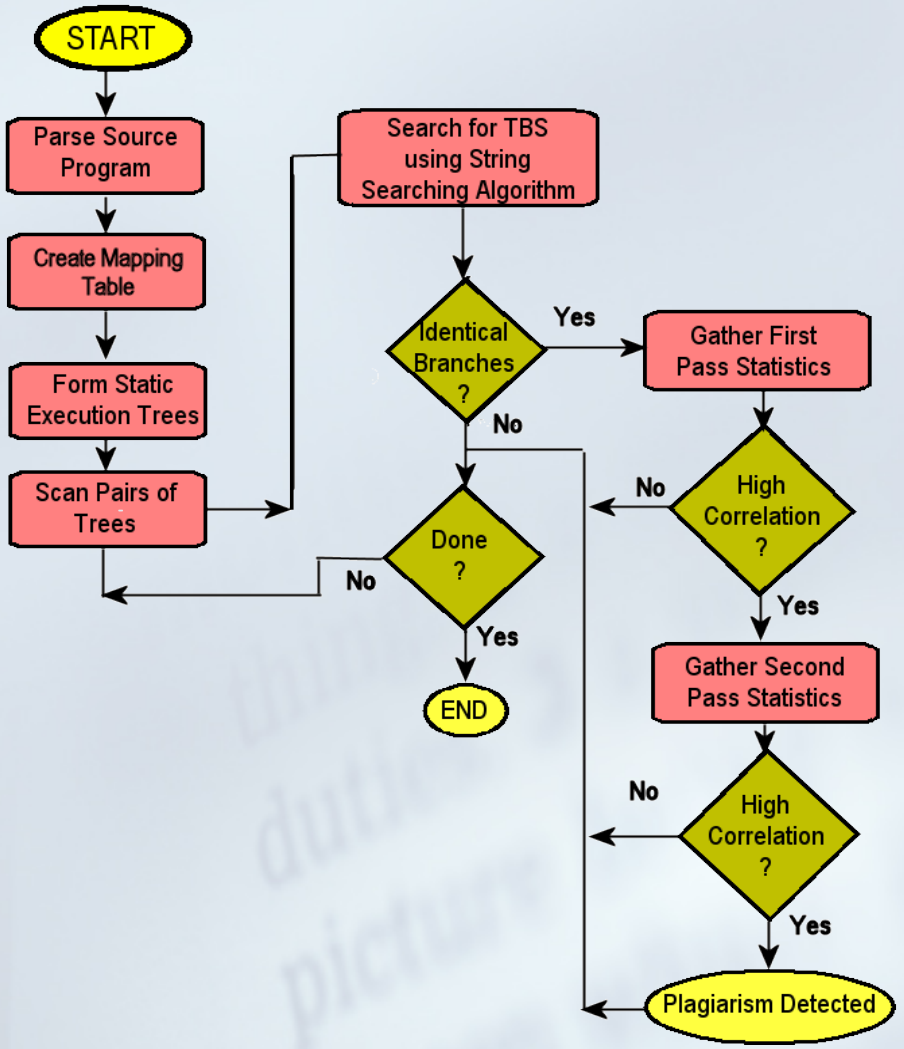
Agenda

- Software Plagiarism Detection
- Defining Source Code
- Software Correlation
- Tools
- Q & A

SOFTWARE PLAGIARISM DETECTION

Plagiarism Measurement

H. T. Jankowitz, 1988, "Detecting plagiarism in student Pascal programs," *Computer Journal*



Plagiarism Measurement

- *Random House Unabridged Dictionary.* (2006). Random House, Inc.

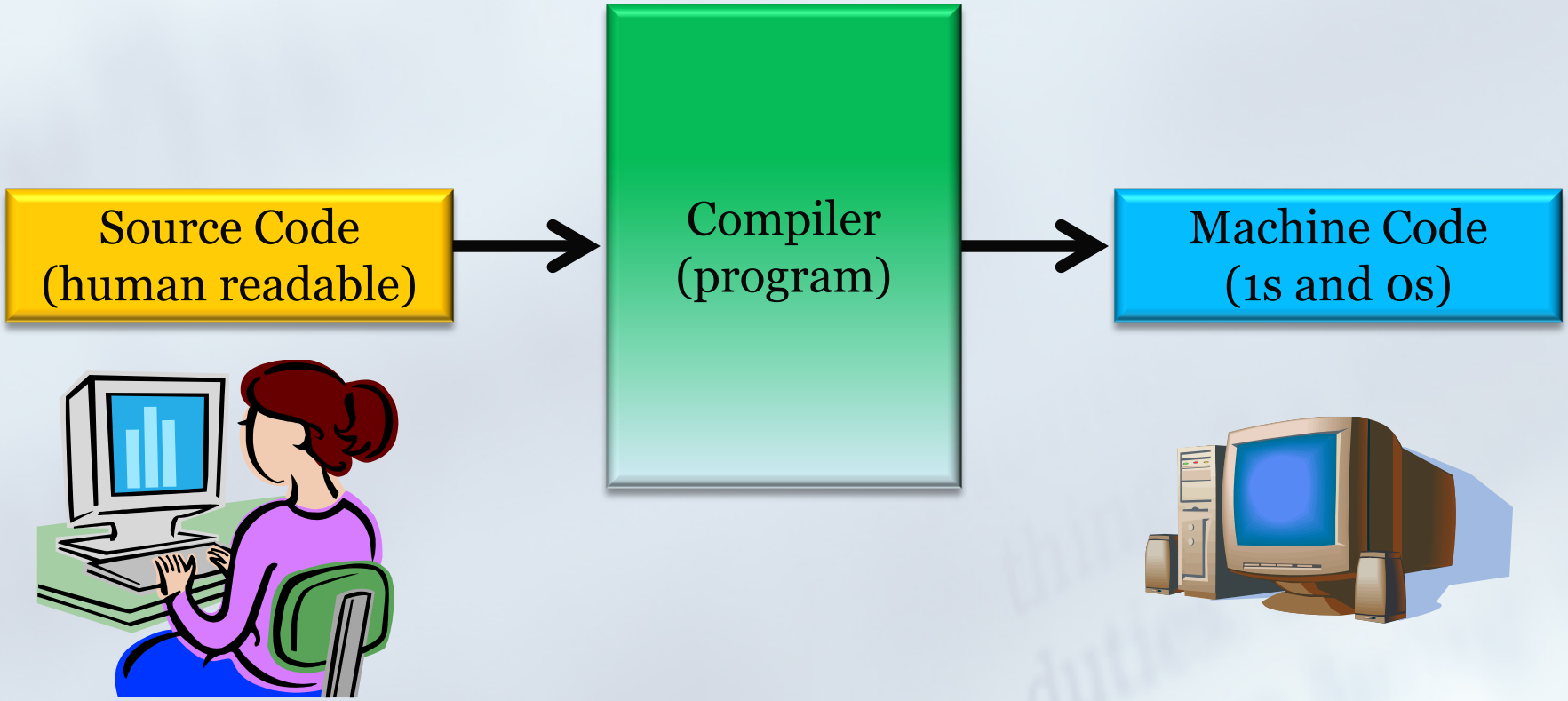
  the unauthorized use or close imitation of the language and thoughts of another author and the representation of them as one's own original work.

Plagiarism Detection

- No definition
- No references
- No standards
- No theoretical basis
- Often reflect the creator's bias
- Need an all-encompassing metric

DEFINING SOURCE CODE

Defining Source Code



Defining Source Code Elements

- Statements: Cause actions, sequential
  - Instructions: Signify the actions to take place.
    - Control words: Control the program flow
    - Specifiers: Specify data allocations or compiler directives
    - Operators: Manipulate data (e.g., +, -. *. /)
  - Identifiers: Reference code or data
    - Variables: Identify data
    - Constants: Identify constants
    - Functions: Identify code
    - Labels: Specify locations in the program
- Comments: Documentation
- Strings: User messages

Defining Source Code

```
// Skip null lines
if (InputLine != NULL)
{
   printf("Store the input line so we can tear it up");

   strcpy(TempLine, InLine);

   InLine [0] = '\0';

   InputIdentifier = strtok (TempLine, SepString);
   while (InputIdentifier != NULL)
   {
      // Put a single space between identifiers
      if (!FirstIdentifier)
         strcat(InLine, " ");
      else
      {
         // Eliminate leading whitespace
         FC = 0;
         while (strchr(SepString, InLine[FC]) != NULL)
            FirstChar++;

         for (i = FC; i <= strlen(InputLine); i++)
            InputLine[i-FC] = InputLine[FC];

         FirstIdentifier = FALSE;
      }
}
```

Source code elements
Statements
Instructions
Control words
Operators
Identifiers
Variables
Constants
Functions
Labels
Comments
Strings

SOURCE CODE CORRELATION

Define Correlation

- 0 for unrelated source code
- 1 for perfectly related source code
- Exact match (reducing whitespace)
- Partial match
- Functional match
- Transformational match

Source Code Correlation

- ρ_S Statement correlation
- ρ_C Comment/string correlation
- ρ_I Identifier correlation
- ρ_Q Instruction sequence correlation
- ρ Overall source code correlation

- μ Match score (unnormalized correlation)

Axioms

- 1. Commutativity $u_X\left(F^n, F^m\right) = u_X\left(F^m, F^n\right)$
- 2. Identity $u_X\left(F^n\right) = u_X\left(F^n, F^n\right)$
- 3. Correlation $\rho_X\left(F^n, F^m\right) = \dfrac{u_X\left(F^n, F^m\right)}{u_X^{\max}\left(F^n, F^m\right)}$
- 4. Maximum match score

$$u_X^{\max}\left(F^n, F^m\right) = \min\left(u_X\left(F^n\right), u_X\left(F^m\right)\right)$$

Correlations

- Statement Correlation
  - The result of comparing functional lines of source code
- Comment/String Correlation
  - The result of comparing non-functional lines of source code
- Identifier Correlation
  - The result of comparing identifiers in the source code
- Instruction Sequence Correlation
  - The result of comparing the sequence of instructions in the source code
- Overall Source Code Correlation
  - Each element correlation can be considered a single dimension that can be used to calculate a multi-dimensional overall correlation.

Correlation Equations

- W-Correlation

$$\rho = \frac{\kappa_S \rho_S + \kappa_C \rho_C + \kappa_I \rho_I + \kappa_Q \rho_Q}{\kappa_S + \kappa_C + \kappa_I + \kappa_Q}$$

- A-Correlation

$$\rho = \frac{1}{4}\left(\rho_S + \rho_C + \rho_I + \rho_Q\right)$$

- M-Correlation

$$\rho = \max\left(\rho_S, \rho_C, \rho_I, \rho_Q\right)$$

- S-Correlation

$$\rho = \frac{1}{2}\sqrt{\left(\rho_S\right)^2 + \left(\rho_C\right)^2 + \left(\rho_I\right)^2 + \left(\rho_Q\right)^2}$$

TOOLS: DETECTING COPYRIGHT INFRINGEMENT

JPlag

- University of Karlsruhe - Guido Malpohl
- Available online (https://www.ipd.uni-karlsruhe.de/jplag)
- Measures sequence correlation ρ_Q
- Ignores identifier correlation ρ_I
- Ignores statement correlation ρ_S
- Ignores comment/string correlation ρ_C

Measure of Software Similarity (MOSS)

- Stanford University – Alex Aiken
- Available online (http://theory.stanford.edu/~aiken/moss)
- Creates a statistical fingerprint
- Treats software like generic text
- Measures some identifier correlation ρ_I
- Measures some statement correlation ρ_S
- Ignores sequence correlation ρ_Q
- Ignores comment/string correlation ρ_C

CodeSuite®

- Software Analysis & Forensic Engineering
- Available online (www.safe-corp.com)
- CodeMatch®
  - Measures full correlation
  - Produces detailed reports
  - Allows filtering
  - Produces statistics spreadsheets

Detecting Copyright Theft

Source Code 1

Source Code 2

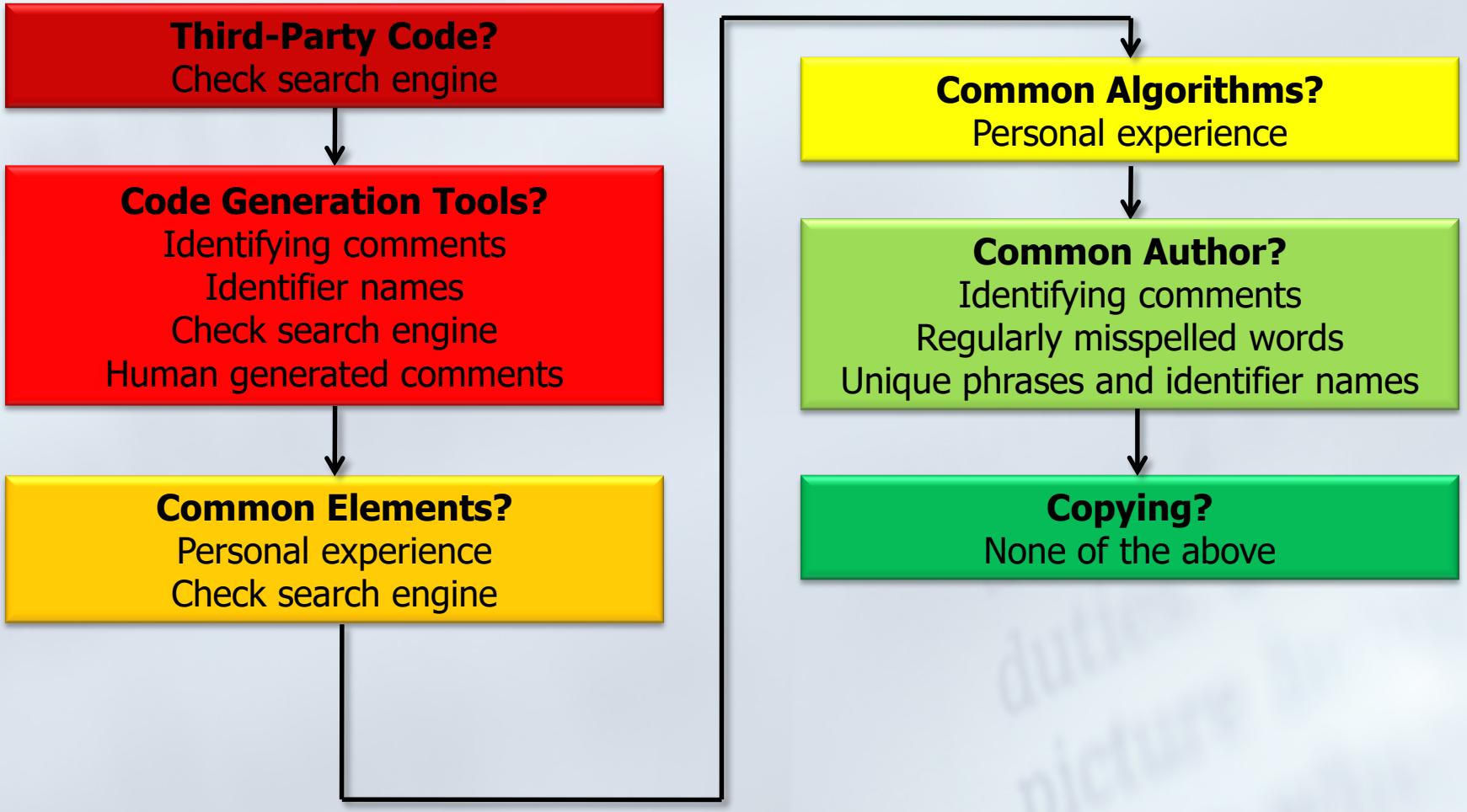
Measure Correlation

Correlation

Reasons for Correlation

- Third-Party Source Code
- Code Generation Tools
- Commonly Used Identifier Names
- Common Algorithms
- Common Author
- Copying (Plagiarism, Copyright Infringement)

Finding Correlation Reason



QUESTIONS AND ANSWERS

Thank You

Bob Zeidman
bob@SAFE-corp.com